Imperial College London

A brief introduction to numerical simulation & Fluidity

Matthew Piggott
m.d.piggott@imperial.ac.uk

Fluidity training workshop Nov 2012

This presentation

- An overview of Fluidity
- Some very brief comments on just some of the things to consider when conducting numerical simulations (of any type – not Fluidity-specific)
- Issues you need to think about
- Things you need to be wary of

- This is a personal view of this area

- Remember that many of the concepts related to stability, convergence, order, best practice, stabilisation, etc, are common to many methods / codes

The Name

- Fluidity, ICOM, Fluidity-ICOM?
- For a number of reasons, some good, some perhaps not so good, a number of names have been used for the code we're discussing here. They are all, and always have been, one and the same thing!
- If an application was CFD-like we tended to use 'Fluidity'
- If an application was GFD/ocean-like we tended to use 'ICOM' (Imperial College Ocean Model)
- Now that the code is released to multiple communities we need to clarify things
- The current convention is to use 'Fluidity' as the name of the CFD model as well as the overall modelling (code) framework
- and sometimes we use Fluidity-ICOM for GFD / ocean applications
- and we may start using Fluidity-**** for other application areas under development (e.g. **** could take the form: Stokes, Rocks, Solids, Atmos, ICAM, RT, ... in the future, yet to be decided) - if in doubt just use 'Fluidity'

What is Fluidity?

- A whole bunch of Fortran, C/C++, Python, etc*
- linked to a number of external libraries, e.g. PETSc (NB. we need to worry about license compatibilities)
- to solve a variety of PDE systems
- using control volume / finite element (CV/FEM) discretisation methods
- on structured and unstructured (possibly adapting) meshes
- in parallel
- coupled to external models / codes as well as internal physics/chemistry/biology/sediment/turbulence modules
- NB. Remember that a fixed structured mesh is a sub-case of an adapting unstructured mesh, but our implementation of CV/FEM methods always assumes unstructured – this has implications on coding and CPU costs

*next slide

Analysis of lines of code in Fluidity + longtests

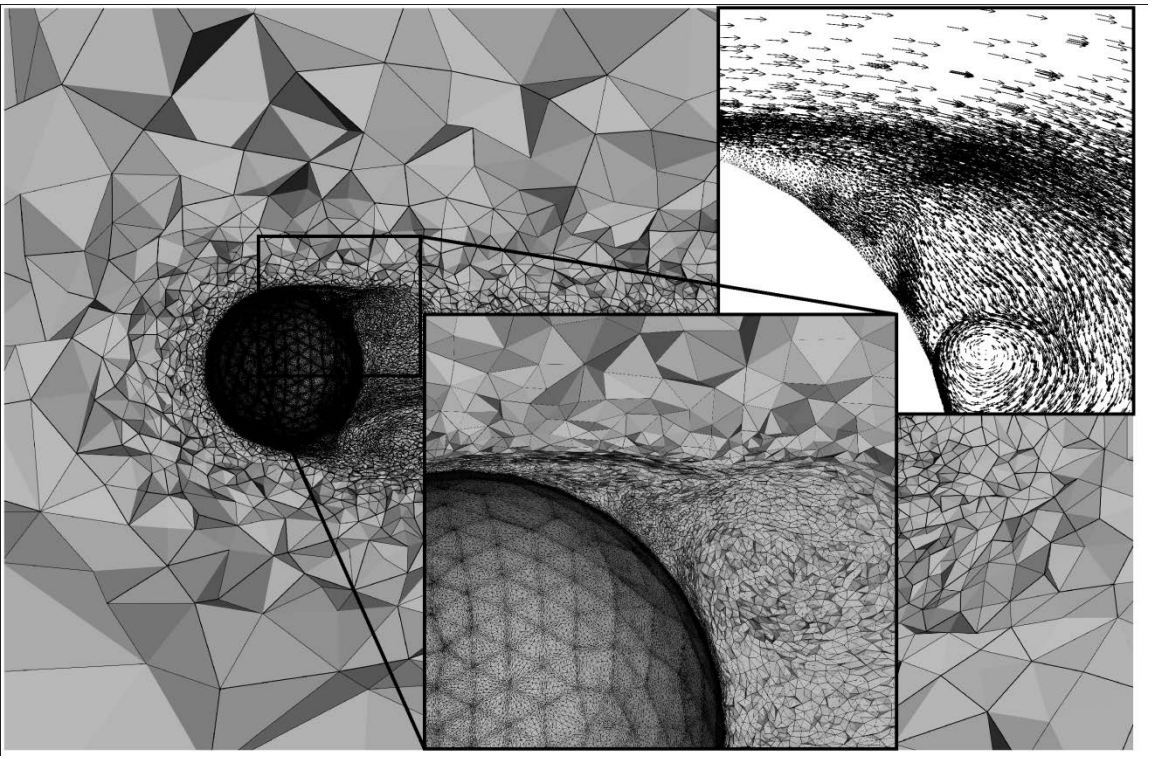
(values from Nov 2011)

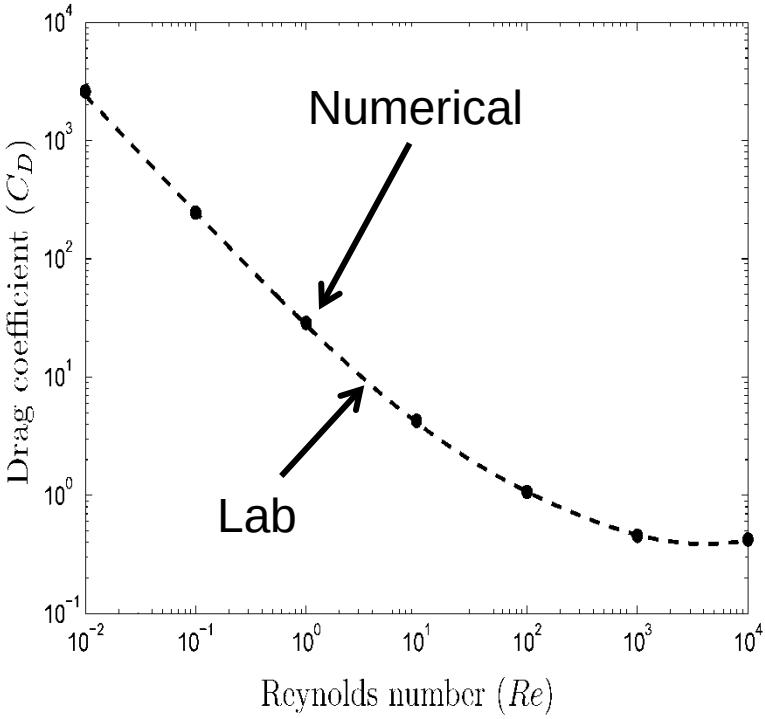
```
5998 text files.
4950 unique files.
1963 files ignored.

http://cloc.sourceforge.net v 1.09  T=8.0 s (432.5 files/s, 124784.9 lines/s)
```

Language	files	blank	comment	code	scale	3rd gen. equiv
FLML Test/example problem description	795	1748	0	354854	x 1.00 =	354854.00
Fortran 90	575	45207	33366	156702	x 1.00 =	156702.00
GEO Describe geometry/mesh	337	338	698	60384	x 1.00 =	60384.00
XML Run and V&V checks for tests	446	4589	2000	44773	x 1.90 =	85068.70
C++	171	10143	7675	43857	x 1.51 =	66224.07
Fortran 77	213	9630	19302	42859	x 0.75 =	32144.25
C	60	7970	8845	27017	x 0.77 =	20803.09
Python E.g. post-processing	245	7403	6203	24653	x 4.20 =	103542.60
RNC Rules for FLML options	26	715	9258	20257	x 1.00 =	20257.00
LATEX Manual	32	3258	632	16833	x 1.00 =	16833.00
C/C++ Header	152	3916	6106	11392	x 1.00 =	11392.00
make	408	879	122	4695	x 2.50 =	11737.50
SUM:	3460	95796	94207	808276	x 1.16 =	939942.21

Some computational fluid dynamics (CFD) examples

CFD **verification/validation**: Comparisons between drag calculation in flow past a sphere at a range of Re numbers





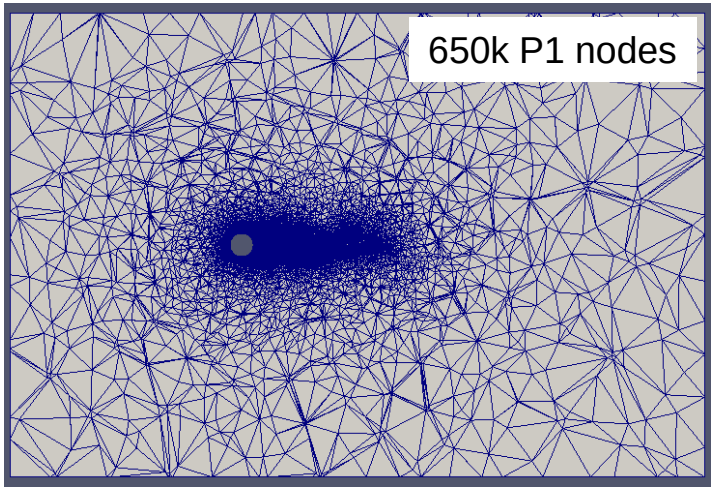
Computed drag coefficient compared against correlation (from Brown and Lawler, 2003) with lab data:

$$C_D = \frac{24}{Re}\left(1 + 0.15Re^{0.681}\right) + \frac{0.407}{1 + \frac{8710}{Re}}$$

Cf. 'flow past sphere' examples in manual/code

A movie showing the magnitude of velocity in a slice through the 3D domain

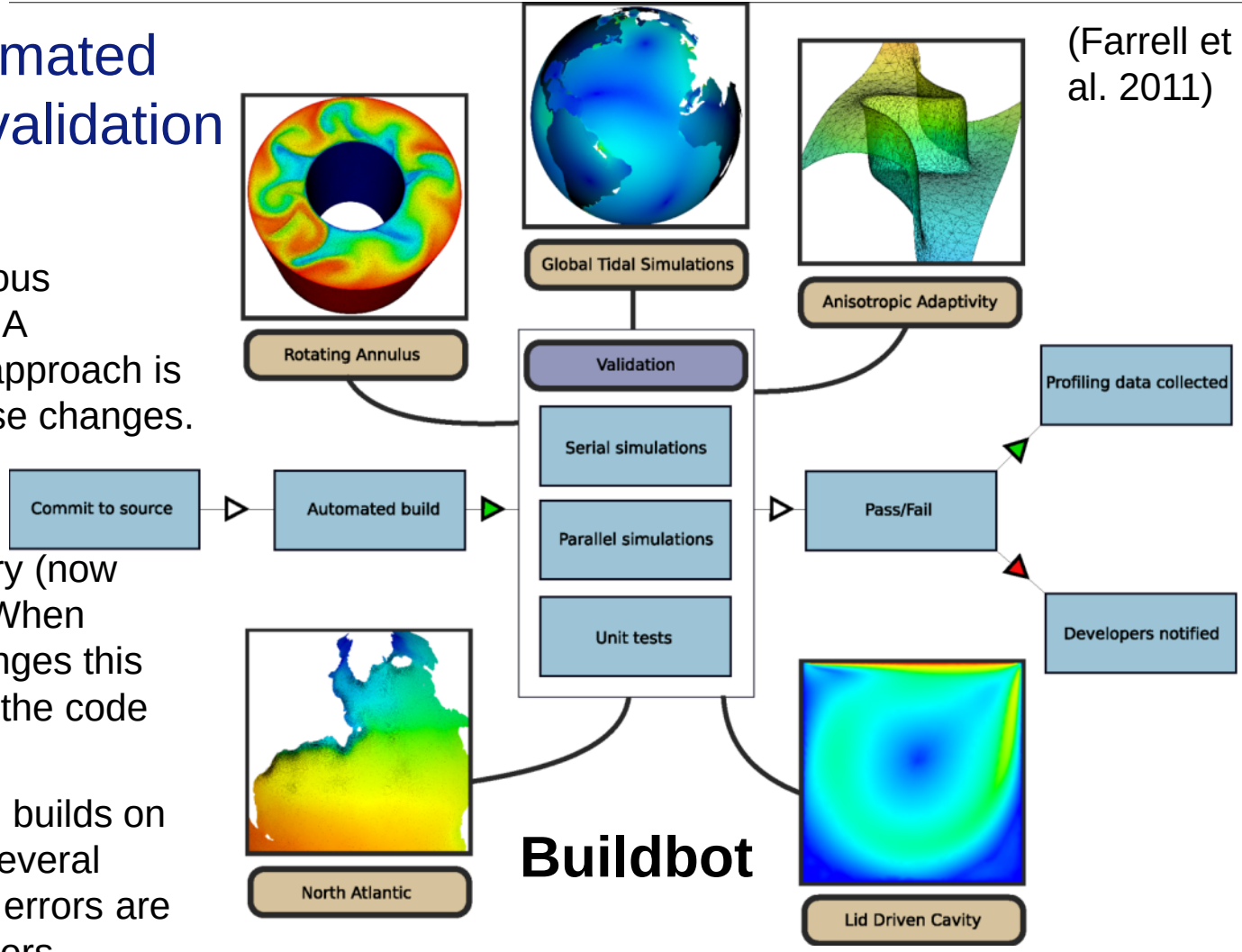
Notice that the problem is using an adaptive mesh and with the settings used the wake becomes under-resolved downstream of the sphere – we don't really care for drag calculaton and have chosen an error metric to guide adaptivity appropriately



Model verification

Robustness: automated code verification/validation system (Buildbot)

(Farrell et al. 2011)



- All models require rigorous verification / validation. A continuous automated approach is required as the codebase changes.
- A central copy of the source is kept in a version control repository (now hosted on launchpad). When developers commit changes this must result in a pass of the code test suite.
- Buildbot checks out and builds on various platforms with several different compilers. Any errors are relayed back to developers.
- If a failure is detected in a test problem, the developers are notified details via email.
- Statistical information about code quality is automatically collected from the newly validated code. This allows for the monitoring of performance. Results available via a web interface.
- This modern approach to software engineering has yielded dramatic improvements in code quality and programmer efficiency.

The method of manufactured solutions (e.g. Salari & Knupp 2000, Roache 2002)

- Problems with analytical solutions are ideal for verifying code
- However, for more complex equations sets (e.g. those with nonlinearities) it is hard to come up with analytical solutions
- The method of manufactured solutions is a clever approach that does things in the opposite way: you start with a function and then find the PDE problem that this is the analytical solution to
- To do this we substitute this function into the PDE system we're interested in, there will be a residual that we can compute using symbolic math packages
- We simply use this residual as a *source term* in the PDE and solve this new system numerically
- The initially chosen function is the analytical solution we compare against, check orders of convergence etc.

See: Roache, P. J.: Code Verification by the Method of Manufactured Solutions, J. Fluid. Eng. T. ASME, 124, 4–10, doi:10.1115/1.1436090, 2002; Salari, K. and Knupp, P.: Code Verification by the Method of Manufactured Solutions, Tech. Rep. SAND2000-1444, Sandia National Laboratories, Albuquerque, NM, 2000.

The method of manufactured solutions – a simple example

To test tracer advection-diffusion the desired analytical solution is taken as:

$$T(x,y,t) = \sin(25xy) - 2y/x^{1/2}, \quad (1)$$

while a prescribed velocity field, $\boldsymbol{u} = (u, v)$, is given by

$$u = \sin\left(5(x^2+y^2)\right), \quad v = \cos\left(3(x^2-y^2)\right). \quad (2)$$

The source term, S, is calculated symbolically using SAGE (Stein et al., 2007) by substituting T and $\boldsymbol{u}$ into the advection-diffusion equation:

$$\begin{aligned} S &= \frac{\partial T}{\partial t} + \boldsymbol{u}\cdot\nabla T - \kappa\nabla^2 T, \\ &= \left(25y\cos(25xy) + y/x^{3/2}\right)\sin\left(5(y^2+x^2)\right) \\ &\quad + \left(25x\cos(25xy) - 2/x^{1/2}\right)\cos\left(3(x^2-y^2)\right) \\ &\quad + \kappa\left(625(x^2+y^2)\sin(25xy) + 3y/(2x^{5/2})\right). \end{aligned}$$

Choose a function to act as our analytical solution

Choose the velocity (and a constant viscosity) to fully describe this PDE system

Substitute these into the PDE and calculate a residual

Cut and paste Python form of this into model as a source term to the PDE

Taken directly from Farrell et al. 2011: Automated continuous verification and validation for numerical simulation, Geoscientific Model Development 4, 435-449, 2011. doi:10.5194/gmd-4-435-2011

The method of manufactured solutions – using a symbolic calculus package such as sage (http://www.sagemath.org/)

```
sage: T(x,y,t,kappa) = sin(25*x*y) - 2*y/sqrt(x)
sage:
sage: T_t = diff(T,t)
sage: T_x = diff(T,x)
sage: T_y = diff(T,y)
sage:
sage: T_xx = diff(T_x,x)
sage: T_yy = diff(T_y,y)
sage:
sage: u(x,y,z,t) = sin(5*(x^2+y^2))
sage: v(x,y,z,t) = cos(3*(x^2-y^2))
sage:
sage: u_x = diff(u,x)
sage: u_y = diff(u,y)
sage: v_x = diff(v,x)
sage: v_y = diff(v,y)
sage:
sage: S = T_t + (u*T_x + v*T_y) - kappa*(T_xx + T_yy)
sage:
sage: S
(x, y, kappa, t, z) |--> (25*y*cos(25*x*y) + y/x^(3/2))*sin(5*x^2 + 5*y^2)
+ (25*x*cos(25*x*y) - 2/sqrt(x))*cos(3*x^2 - 3*y^2) + 1/2*(1250*x^2*sin(25*x*y)
+ 1250*y^2*sin(25*x*y) + 3*y/x^(5/2))*kappa
```

Then cut and paste analytical expressions for source terms, boundary terms, and any prescribed velocities etc into Fluidity's diamond GUI directly as Python code

Check convergence on a series of mesh resolutions (cf. Farrell et al. 2011 paper)

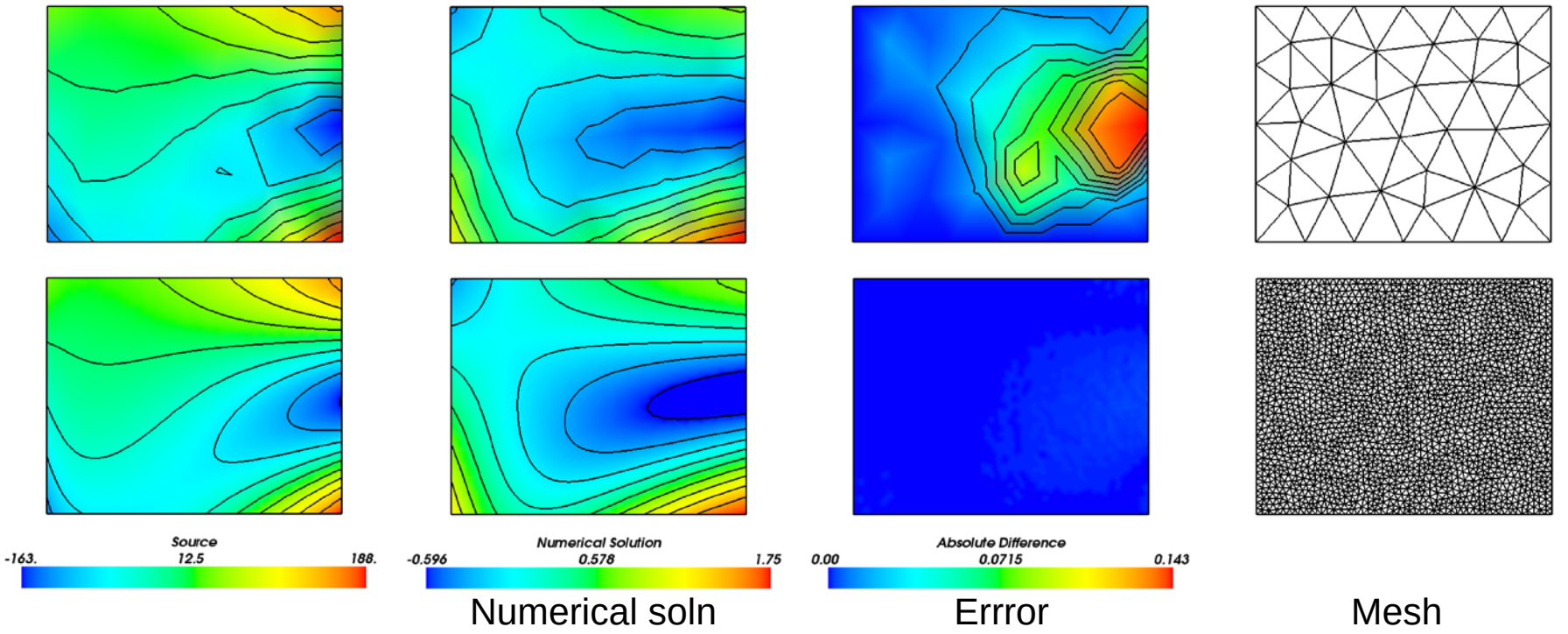


Fig. 3. From left to right: source term (Eq. 3) for the method of manufactured solutions advection-diffusion test case, the numerical solution calculated using a piecewise-linear Galerkin discretisation, the absolute difference between the analytical and numerical solutions, the meshes used to compute the previous images with average mesh spacings, h, of 0.08 (top) and 0.01 (bottom).

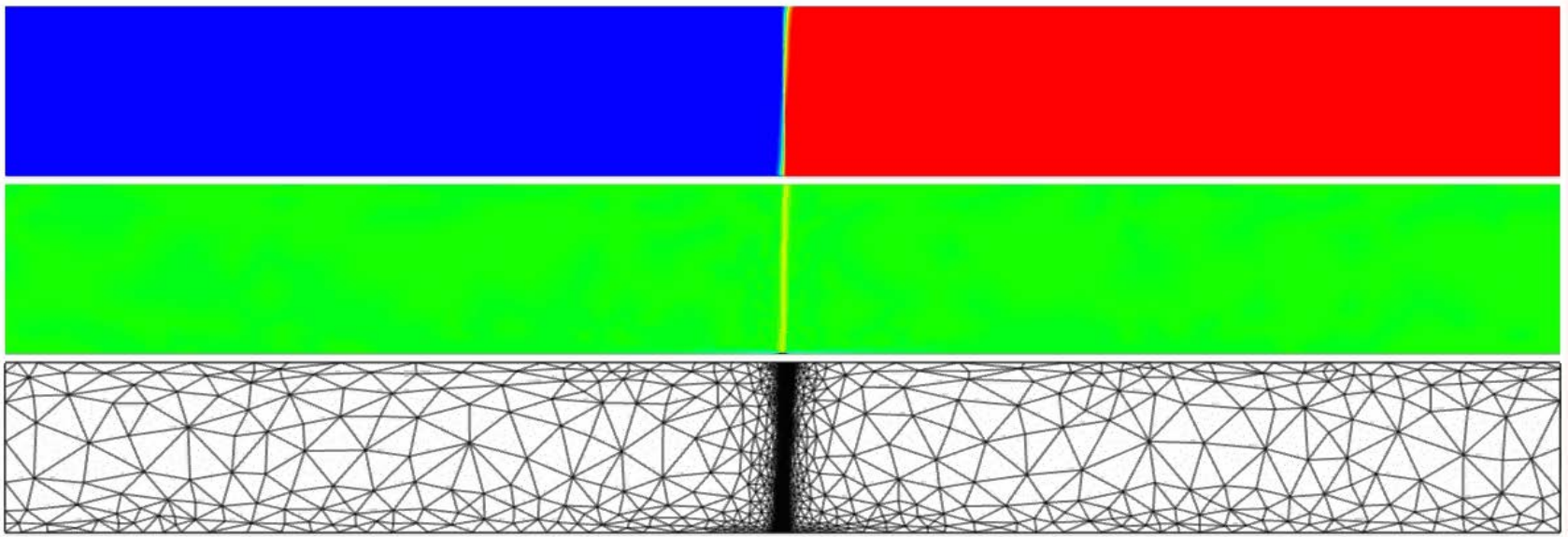
	$h_1 \rightarrow h_2$	$0.08 \rightarrow 0.04$	$0.04 \rightarrow 0.02$	$0.02 \rightarrow 0.01$	$0.01 \rightarrow 0.005$
First-order CV method	c_p (CV)	2.42	2.00	1.43	0.97
Second-order CG method	c_p (P1)	2.03	1.91	2.08	2.12

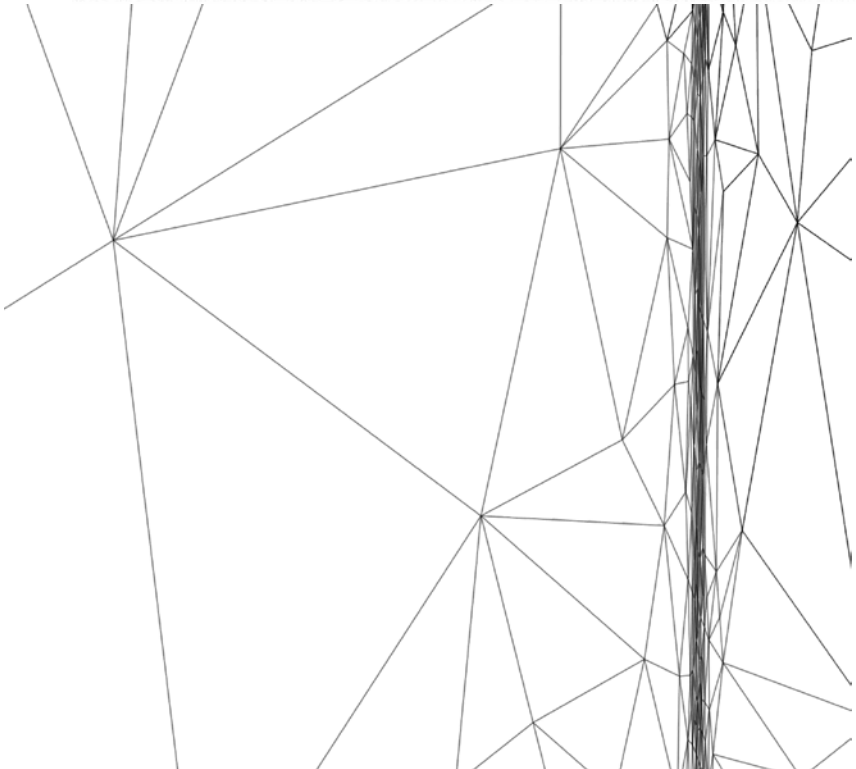
NB. For a second-order method the error should reduce by a factor 4 when halving element size

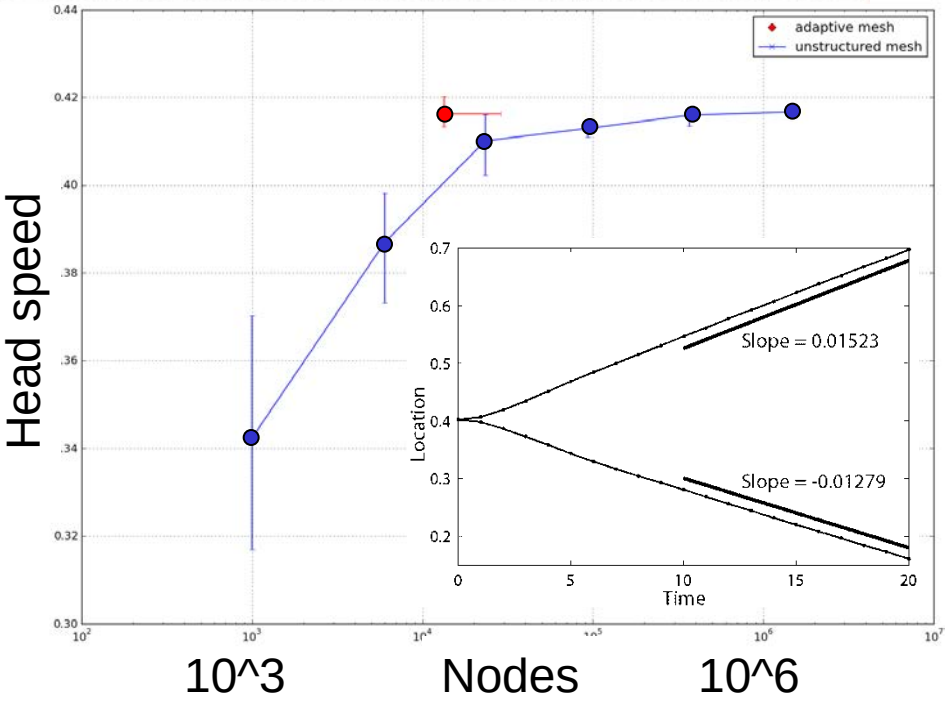
Table 1. Spatial order of convergence results for the method of manufactured solutions advection-diffusion test case described in Sect. 4.1.1. The difference between the analytical and numerical solutions using a first-order control volume (CV) discretisation and a second-order piecewise-linear Galerkin (P1) discretisation are calculated in the L_2 norm. The ratio between these on two spatial mesh resolutions, h_1 and h_2, are used to estimate the order of spatial convergence of the model for this problem. The expected order of convergence, or better, is observed for both spatial discretisations.

Some geophysical fluid dynamics (GFD) examples

2D example with buoyancy: lock exchange problem and front speed validation

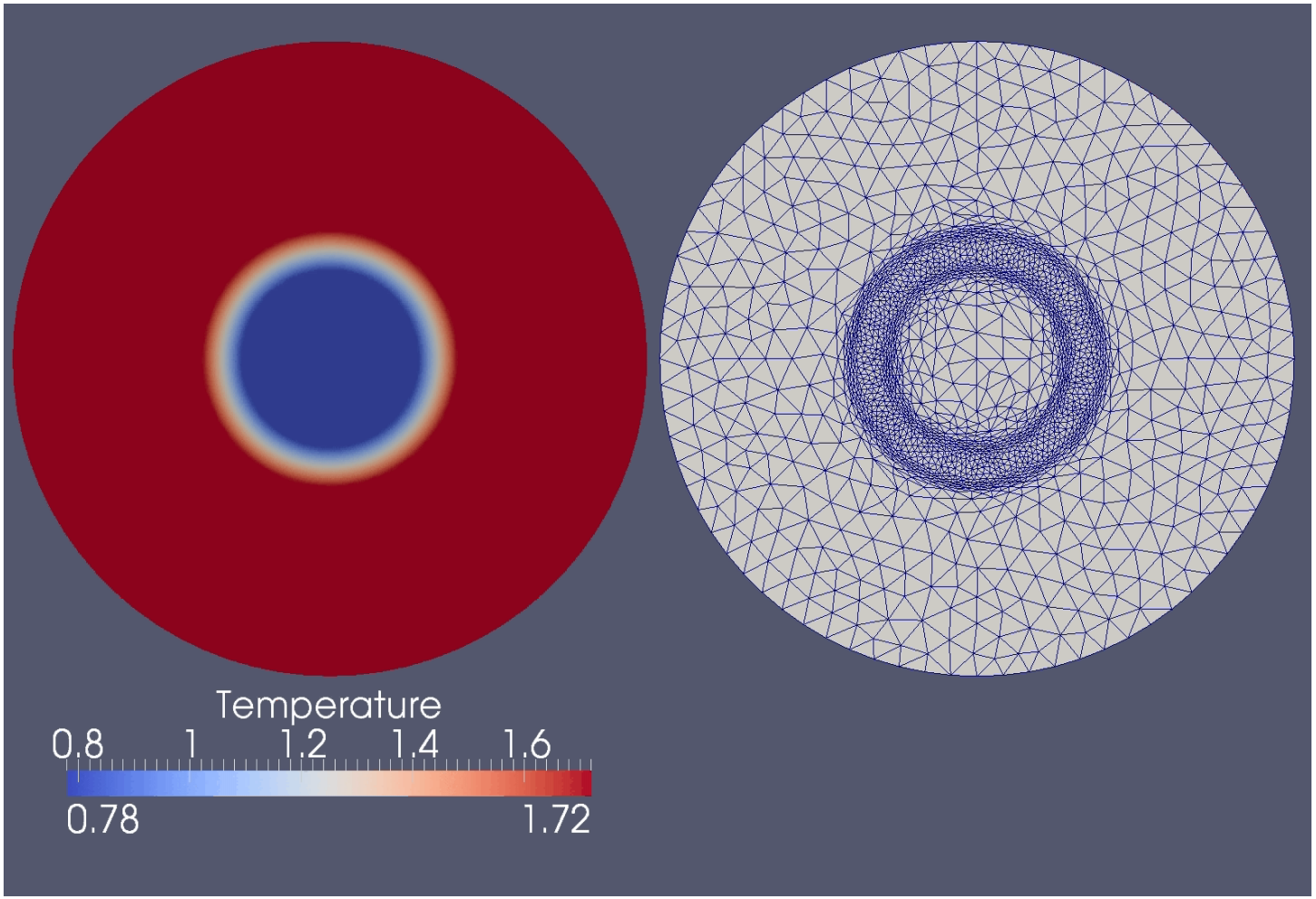


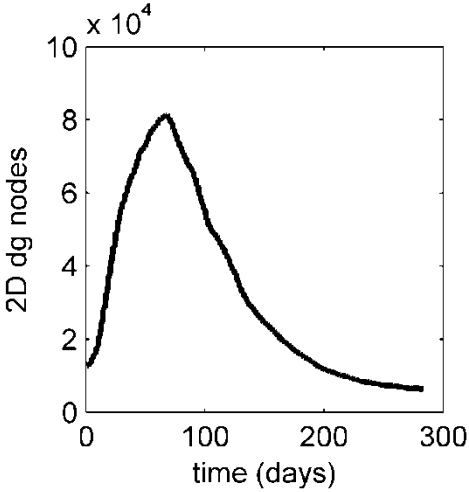




Hiester HR, Piggott MD, Allison PA, The impact of mesh adaptivity on the gravity current front speed in a two-dimensional lock-exchange, Ocean Modelling 38, 1-21, 2011.

The surface T field and the 2D adapted surface mesh (NB. this is using 2+1D adaptivity)





Highly multi-scale tides: marine renewables applications

Some introductory comments on numerical simulation

What should you worry about when choosing which model to use?

- Does it claim to model the sort of **problem / physics you want**?
- Has it **demonstrated** this claim?
- Do you have to **pay** for it?
- Does it make use of pre/post-processing **software that you have access to**?
- Is it **easy to set-up problems** from scratch, how many useful sample configurations / examples are there?
- Is there a **testing suite that ensures code robustness**?
- Can you see the **source code**, and edit / **contribute** if you want?
- Will you be able to **compile** and usefully run it on the computers you have access to?
- How **active** is **development** and what is the likely **longevity** of the project?

Some important things to consider in numerical simulation

(physical considerations)

- What is your **domain** / geometry?
- What are the underlying **physics** (dominant processes, physical parameters, non-dimensional numbers, etc)?
- What **equations** should you consider – can they be solved accurately and stably in the parameter regimes you are interested in?
- What are the **boundary conditions**? Are they consistent with the physics / equations?
- What are the **initial conditions**? Are they consistent with boundary conditions, a divergence-free initial velocity etc?
- **Forcing** – do you need an external data set or can you set via an analytical function?

Some important things to consider in numerical simulation
(numerical considerations)

- **Mesh** – start simple – can you use a uniform, or structured mesh (initially at least)? And start coarse – if possible, so you get fast run times and then finesse your set-up once you're happy there are no fundamental problems with the set-up
- Numerical **discretisation** – a big topic, some of which covered briefly yesterday
- Do you need to 'model' **unresolved processes** or physics that is not represented (another mammoth topic – turbulence … subgrid scale parameterisation)?
- Is **stabilisation** required? – use with care, possibly start with this to get something working robustly at coarse resolutions and then slowly try removing
- **Outputs / diagnostics** – are they available, are they costly and should you compute them online or offline?
- Qualitative and quantitative **analysis of outputs** … **error** analysis
- Mesh **refinement**, both through finer uniform/non-uniform meshes as well as via mesh adaptivity
- **Convergence** (spatial and temporal)

Further things to consider in numerical simulation

- Remember that a numerical model really is a **model** – it only approximates the real system, and that system may be so complex, or the model poor, that this model may make **huge assumptions / simplifications**
- Remember that **a single simulation is almost certainly useless**,
- as it's quite **possible to get pretty much the right answer for the wrong reasons!**
- You should run an appropriately chosen **suite** of simulations
- You should **vary details of the discretisation and mesh** to check robustness
- and very likely you will want to **vary some of the physical parameters** governing the problem - to understand the sensitivity of your results to these two things
- Think about this when you are **designing** your simulation set-up
- You will have to **be self-adaptive**, e.g. based on your first simulation, and the time it takes to run, you may have to rapidly re-think things

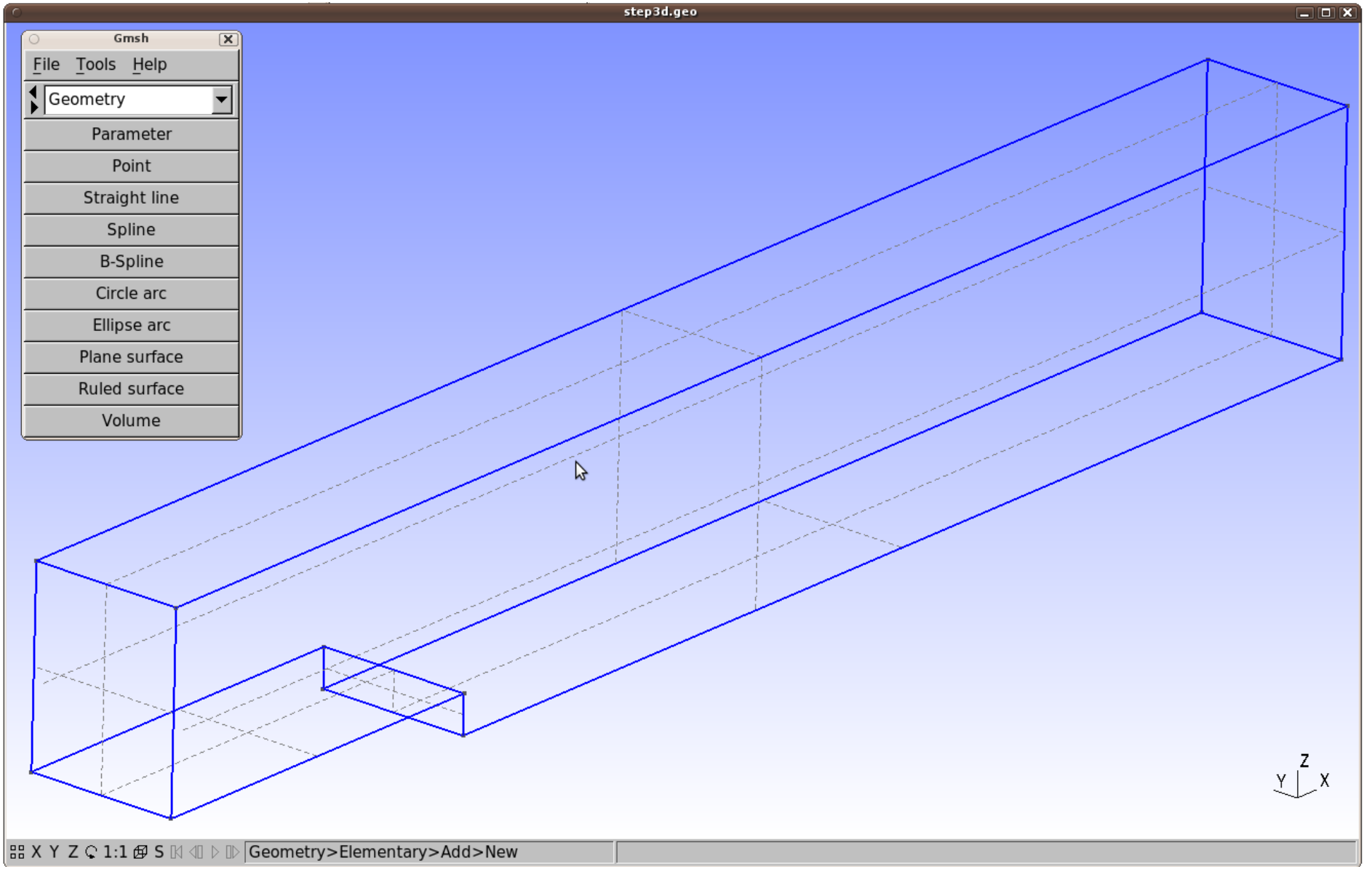
Why worry about these issues?

- In presentations and publication reviews you will almost certainly be asked questions such as:

- “How do you know that your results are correct / trustworthy?”
- “Are your simulations, and hence your conclusions, sufficiently independent of (or robust to) the details of the discretisation and mesh resolution employed?” – if you changed one of these would your conclusions change?

- So think about whether you have been sufficiently rigorous in your experiments (and / or are familiar with other relevant studies that others have performed) to robustly respond to these

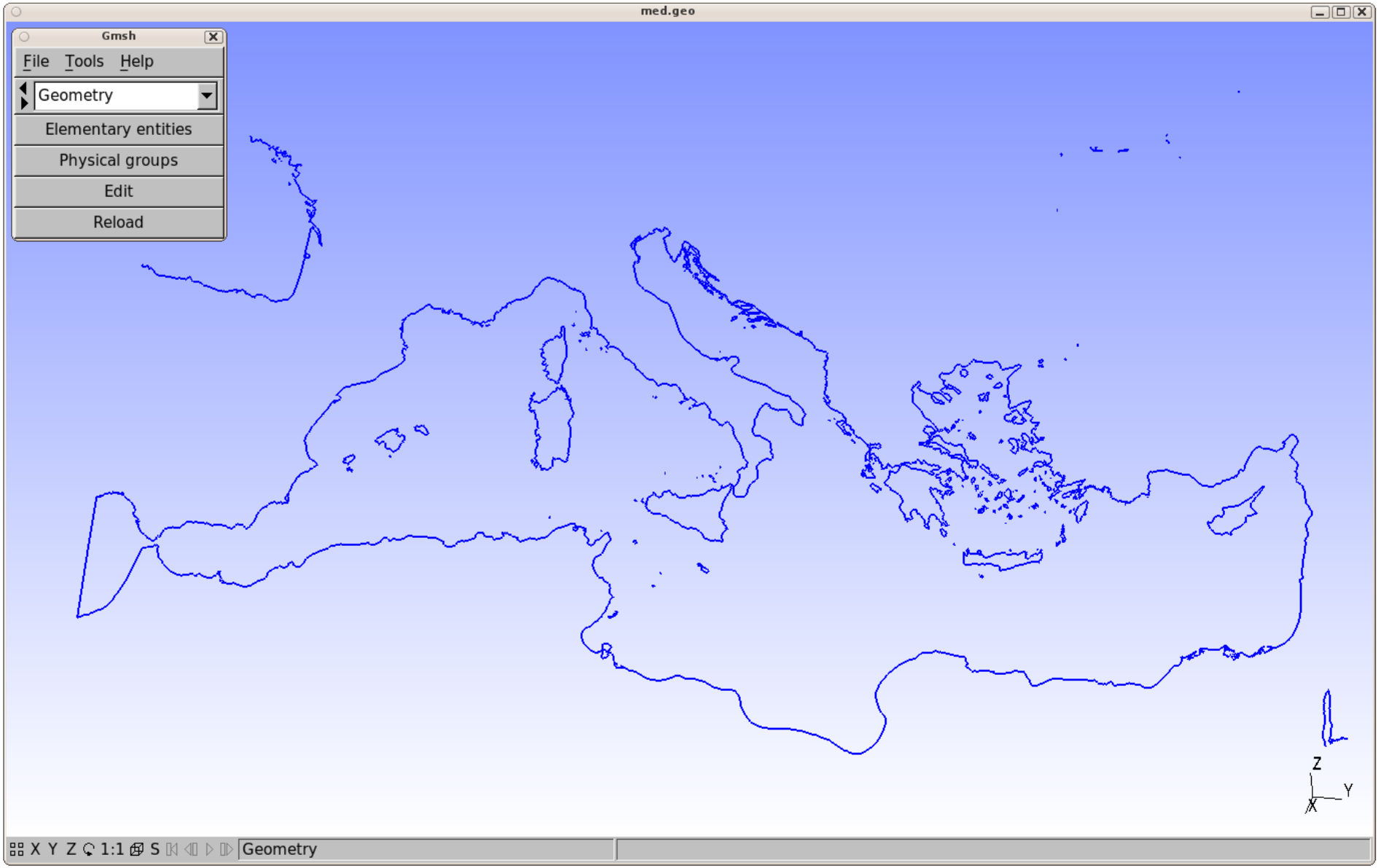
Domain / geometry – one of the CFD examples



Domain / geometry

- Obviously this is critical as it is likely to be a significant controlling factor on the solution - often it will be prescribed but sometimes, e.g. with coastlines, simplifying approximations will have to be made
- Can it be defined simply, e.g. by points and straight lines in Cartesian space?
- If so, a number of mesh generators can be used to represent the surface of the domain and then mesh the interior, our current favourite is Gmsh (open source), of which more in later presentations
- If it is more complex then you may need to interface to a sophisticated CAD package, or a CAD description of the geometry may be your starting point. You may need to invest more time, effort and possibly money in additional mesh generation software
- When designing the geometry you will need to think about the type and number of boundary conditions you will want to apply and consequently apply surface labels in order to identify them later

Domain / geometry – one of the GFD examples



Domain / geometry – some advice

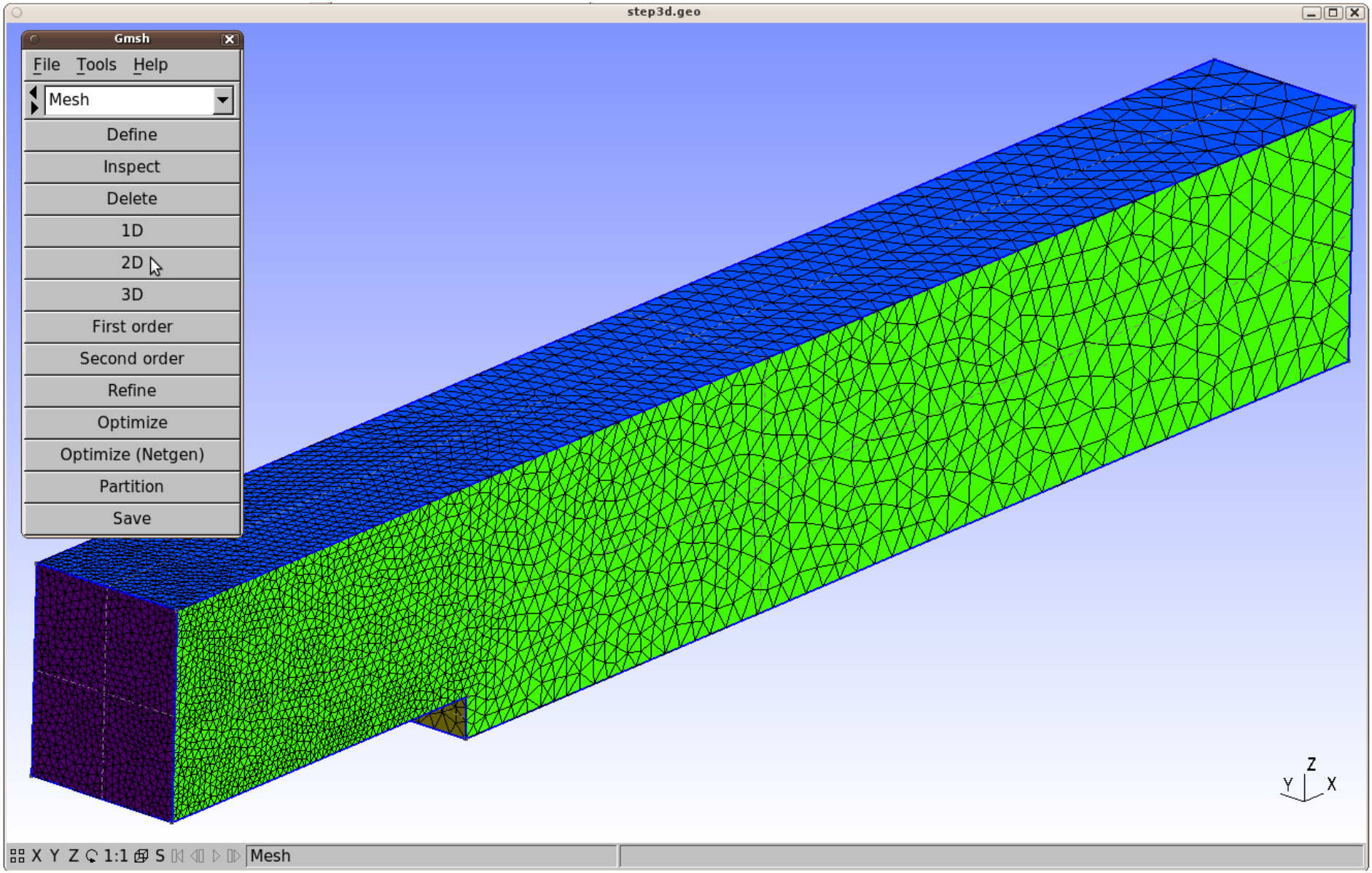
- Start as simple as possible and build up the complexity when you are confident you can run a simple problem in a simplified geometry

- Think ahead – perhaps use more labels to ID surfaces than you think you need right now, this will give you the flexibility to generalise your set-up in the future

- Add comments to the file describing your geometry / surface labelling / meshing options so you, or others, can easily edit later or re-use for a similar problem

- **Generic advice**: it is strongly recommended that you start from something that you know already works, whether this be a mesh file or an flml etc, choose something as close as possible to the scenario you want and make incremental changes!

An unstructured mesh of tetrahedra for a CFD example

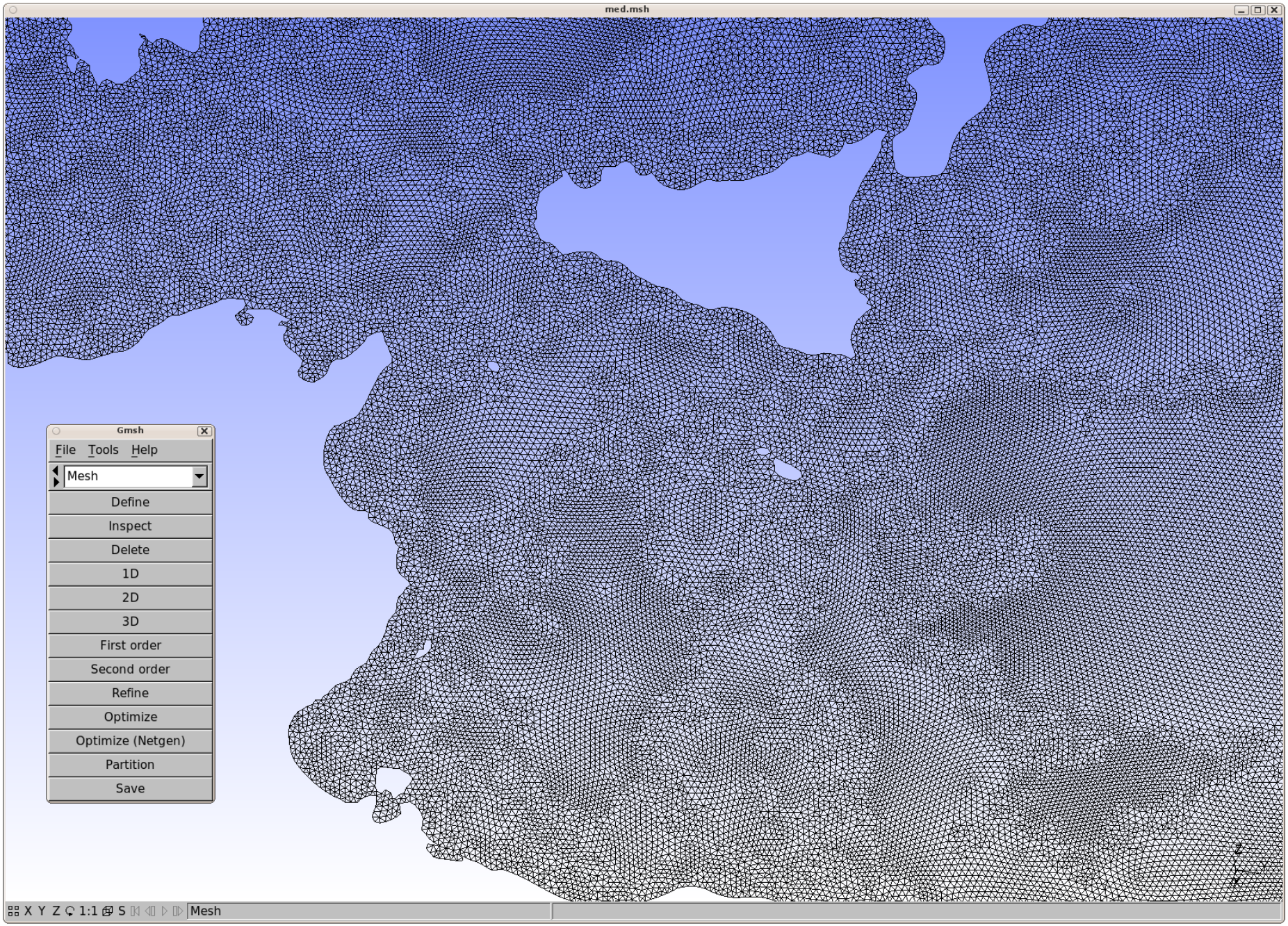


Mesh

- Define the geometry
- Mesh the surfaces (assuming 3D geometry)
- Mesh the volume (assuming 3D geometry)
- Do you want mesh refinement in certain regions?
- Do you want to construct the mesh region-by-region and do you want to preserve this distinction? E.g. to preserve material properties or exactly represent initial conditions or boundary conditions
- Are you going to employ dynamic mesh adaptivity in the simulation?

- Some advice:
  - Start simple, both the geometry and mesh, and then systematically build up complexity
  - e.g. start with a box and a structured uniform mesh
  - Think ahead, choose divisions of lines so you can easily refine/coarsen later in powers of 2, e.g. starting with 2^n divisions gives you scope to both refine and coarsen in a systematic way to look at convergence

An unstructured mesh of triangles for a GFD example



Physics and equations

- What are the dominant physical processes?
- What are the unknowns?
- What equation sets do you need and how are they coupled?
- What are the key physical parameters?

- Some advice:
  - Consider starting simply, e.g. possibly consider not including buoyancy or rotational effects initially and build up the complexity systematically
  - Remember that parameters such as viscosity will have a strong impact on the scales of motion – start with parameters that should yield simpler physics (e.g. a larger viscosity), and hence easier simulation, and then build up the complexity slowly

Boundary conditions (BCs)

- Don’t forget that these are just as important as the underlying equations – the problem is not completely described without appropriate BCs
- They should be based on physics, i.e. they are strongly dependent on the equations being solved and also on the geometry, not all BCs are possible / yield well-posed problems
- Many problems are strongly influenced by behaviour at boundaries and errors here are very likely to pollute the solution everywhere
- Lack of the expected order of convergence could be due to inaccuracies in BCs
- The BCs you can apply, and how they are implemented, is strongly dependent on the discretisation you are using (e.g. with FEM which terms you integrate by parts)
- We are in good shape with FEM as things match the physics naturally, but there are many options for the precise details of the BCs and so lots of scope for going wrong
- You may need to interface to external data sets, e.g. for real world atmospheric forcing

Initial conditions (ICs)

- Should be consistent with the underlying equations and boundary conditions
- Do you know the initial conditions from a previous run or an external data set?
- E.g. you may want to spin-up a base run, checkpoint, and start a suite of experiments from this
- Do you want to define an analytical function, or interpolate from another mesh?
- Can you spin-up some variables from zero? A common choice for velocity

Outputs

- If you start by comparing to another numerical simulation that reports / plots a particular output metric, or a system for which some sort of observational data exists, can you reproduce these quantities for your simulation?

- Is it a costly calculation to compute this quantity – is it best to calculate online or offline – do you need consistency, will you want values at every time step?

- Run a base simulation with all the diagnostic outputs you want, and then without any, note the difference in CPU time, there could be a significant difference and this may make you far more choosy in what outputs you switch on, or to re-design an online quantity to be computed offline

- Remember that taking derivatives of a numerical solution can rapidly erode accuracy and can lead to noisy output diagnostics, that's pretty much a fact of life in numerics

- Do you need the diagnostic to be computed in a consistent manner with the discretisation, e.g. consider which function space it's appropriate to do things in

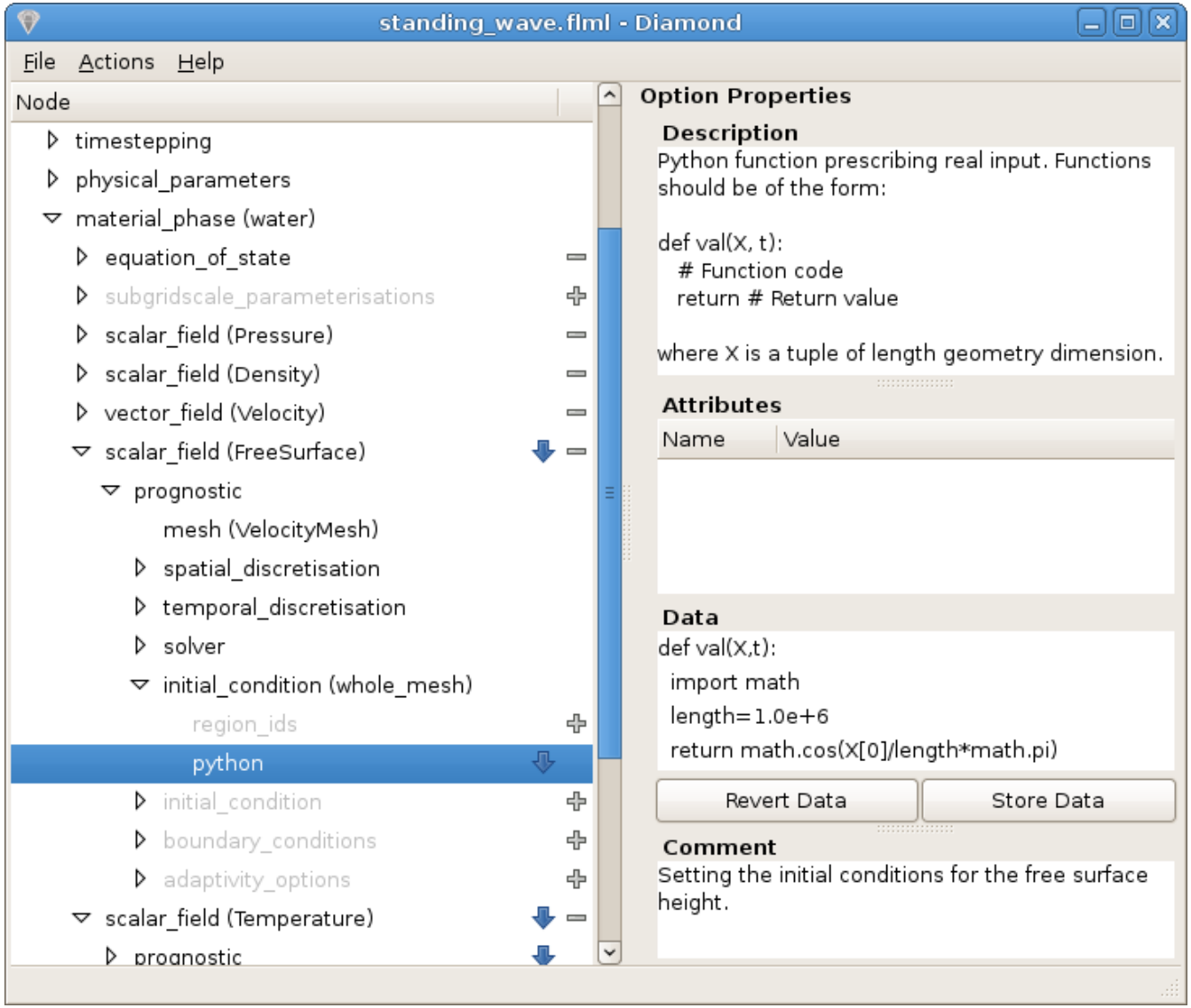
Discretisation options

- This is an entire career in itself – a huge range of options exist
- A large number are available in Fluidity – note that it is far easier to choose a bad combination than a good one! Check the manual
- The ability to make good decisions comes from a large amount of study and experience – look at what is used in Fluidity examples/tests
- Don’t forget that accuracy, stability, appropriateness of certain BCs, are all reliant on the discretisation choices

- Some advice:
  - Ask for advice!
  - Start from a numerical configuration already set-up for a similar problem, but don’t assume this is all appropriate for your problem or even for the problem it previously solved!
  - Try to understand why particular choices have been made - ask lots of questions on why
  - Start defensive – a very safe / boring option with lots of stabilisation and implicitness, high viscosity/small timestep etc, and then finesse/optimise once you have something working
  - Read chapter 3 of the manual, the references therein, and at the end of this presentation, also look at some of the PhD theses on the AMCG wiki

Usability: problem set-up with Diamond (more later today)

- A schema (.rnc) file describes the rules that govern model inputs
- Diamond uses this to automatically generate a GUI based on the schema
- Options are entered and output as another xml file containing the options values (.flml extension)
- This is written to an options library accessible from anywhere in code
- Includes many features, including the ability to define python functions executed at run time



Some recent references

- Cotter CJ, Ham DA, Pain CC, Reich S, LBB stability of a mixed Galerkin finite element pair for fluid flow simulations, J COMPUT PHYS, 2009, Vol:228, Pages:336-348, http://dx.doi.org/10.1016/j.jcp.2008.09.014
- Farrell PE, Piggott MD, Pain CC, Gorman GJ, Wilson CR, Conservative interpolation between unstructured meshes via supermesh construction, Comp. Meth. Appl. Mech. Eng., 198, 2632-2642, 2009. doi:10.1016/j.cma.2009.03.004
- Farrell PE, Piggott MD, Gorman GJ, Ham DA, Wilson CR, Automated continuous verification and validation for numerical simulation, Geoscientific Model Development Discussions, 2010, Vol:3, Pages:1587-1623. http://dx.doi.org/10.5194/gmdd-3-1587-2010
- Stephan C. Kramer, Colin J. Cotter and Chris C. Pain. *Solving the Poisson equation on small aspect ratio domains using unstructured meshes*, submitted to Ocean Modelling. http://arxiv.org/pdf/0912.1976
- Ham DA, Farrell PE, Gorman GJ, Maddison JR, Wilson CR, Kramer SC, Shipton J, Collins GS, Cotter CJ, Piggott MD, Spud 1.0: generalising and automating the user interfaces of scientific computer models, Geosci. Model Dev., 2, 33-42, 2009. http://www.geosci-model-dev.net/2/33/2009/gmd-2-33-2009.html
- Mitchell AJ, Allison PA, Piggott MD, Gorman GJ, Pain CC, Hampson GJ, Numerical modelling of tsunami propagation with implications for sedimentation in ancient epicontinental seas: the Lower Jurassic Laurasian Seaway, Journal of Sedimentary Geology 228, 81-97, 2010. doi:10.1016/j.sedgeo.2010.03.008
- Pain CC, Umpleby AP, de Oliveira CRE, Goddard AJH, Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations, Computer Methods in Applied Mechanics and Engineering, 190 (29-30), 3771-3796, (2001), doi:10.1016/S0045-7825(00)00294-2
- Piggott MD, Gorman GJ, Pain CC, Allison PA, Candy AS, Martin BT, Wells MR, A new computational framework for multi-scale ocean modelling based on adapting unstructured meshes, International Journal for Numerical Methods in Fluids 56, 1003 - 1015, 2008, doi:10.1002/fld.1663
- Piggott MD, Pain CC, Gorman GJ, et al, h, r, and hr adaptivity with applications in numerical ocean modelling, Ocean Modelling 10, 95 - 113, 2005, doi:10.1016/j.ocemod.2004.07.007
- Piggott MD, Farrell PE, Wilson CR, Gorman GJ, Pain CC, Anisotropic mesh adaptivity for multi-scale ocean modelling, Phil. Trans. R. Soc. A 367, no. 1907, 4591-4611, 2009. doi:10.1098/rsta.2009.0155
- Gorman GJ, Pain CC, Piggott MD, Umpleby AP, Farrell PE, Maddison JR, 2009: Interleaved parallel tetrahedral mesh optimisation and dynamic load-balancing, Adaptive Modeling and Simulation 2009, 101-104, Bouillard Ph, Diez P (eds.), CIMNE.
- Wells MR, Allison PA, Piggott MD, Hampson GJ, Pain CC, Gorman GJ, Tidal modelling of an ancient tide-dominated seaway, part 1: model validation and application to global mid Cretaceous (Aptian) tides, Journal of Sedimentary Research 80, 393-410, 2010. doi:10.2110/jsr.2010.044

Some recommended text books

- Elman HC, Silvester DJ, Wathen AJ, Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluid Dynamics , OUP, 2005.
- Donea J, Huerta A, Finite Element Methods for Flow Problems, Wiley, 2003
- Gresho PM, Sani RL, Incompressible Flow and the Finite Element Method (2 volumes), Wiley, 2000
- Zienkiewicz OC, Taylor RL, The Finite Element Method (3 volumes), Butterworth-Heinemann, 2005
- Frey PJ, George P-L, Mesh Generation: Application to Finite Elements, Hermes, 2000